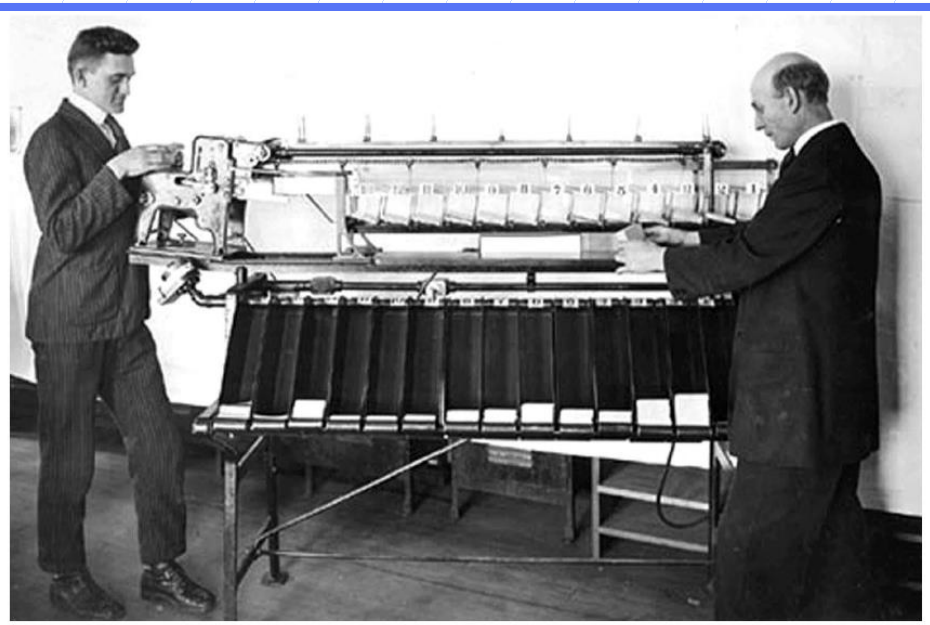
Παρουσίαση για χρήση με το σύγγραμμα, Αλγόριθμοι Σχεδίαση και Εφαρμογές, των M. T. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

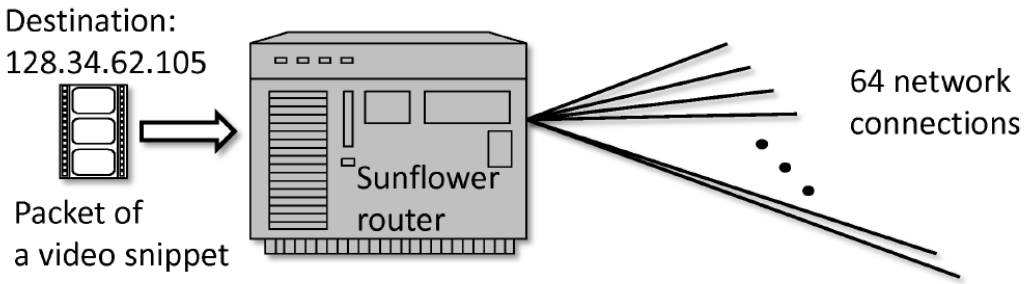
Χάρτες



Operating a card sorter, 1920. U.S. Census Bureau.

Εφαρμογή: Δρομολογητές δικτύου

- Οι δρομολογητές δικτύου επεξεργάζονται πακέτα πληροφοριών από πολλές συνδέσεις σε υψηλή ταχύτητα
- Για την επεξεργασία ενός πακέτου, (k,x), όπου k είναι το κλειδί για τον προορισμό και x τα δεδομένα που περιέχει, ένας δρομολογητής πρέπει πολύ γρήγορα να αποφασίσει σε ποια από τις συνδέσεις του δικτύου να στείλει το πακέτο.
- Ένα τέτοιο σύστημα πρέπει να υποστηρίζει αναζητήσεις βάση κλειδιού, δλδ, λειτουργίες get(k), καθώς και λειτουργίες put(k,c) για την προσθήκη μίας νέας σύνδεσης δικτύου, c, για κλειδί προορισμού, k.
- Ιδανικά θέλουμε να επιτύχουμε O(1) χρόνο τόσο για τη λειτουργία get όσο και για τη λειτουργία put.



Χάρτες

- Ένας χάρτης είναι μία συλλογή με δυνατότητα αναζήτησης για εγγραφές της μορφής κλειδί-τιμή
- Οι κύριες λειτουργίες ενός χάρτη είναι η αναζήτηση, η εισαγωγή, και η διαγραφή στοιχείων
- Πολλαπλές εγγραφές με το ίδιο κλειδί δεν επιτρέπονται
- Άλλες εφαρμογές:
  - Βιβλίο διευθύνσεων
  - Βάση δεδομένων εγγραφών σπουδαστών

Λειτουργίες χάρτη

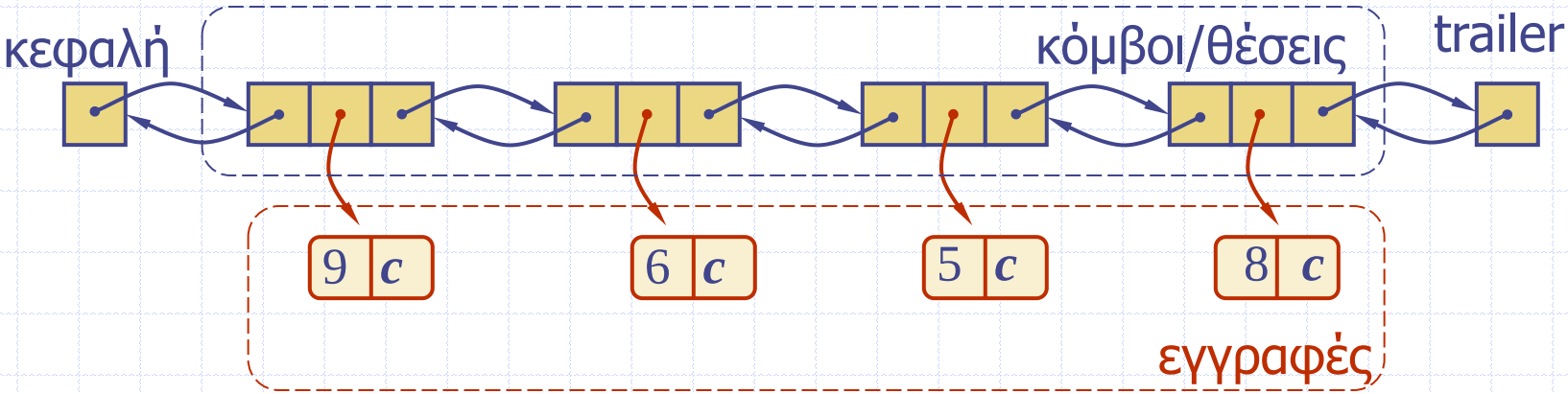
- get(k): Αν ο χάρτης M περιέχει ένα στοιχείο με κλειδί ίσο με k επιστρέφει την τιμή του, αλλιώς επιστρέφει null
- put(k, v): εισαγωγή στοιχείου (k, v) στον χάρτη M, εάν το κλειδί k δεν υπάρχει στον M επιστρέφει null, αλλιώς επιστρέφει την προηγούμενη τιμή στην οποία αντιστοιχούσε το k
- remove(k): εάν ο χάρτης M έχει ένα στοιχείο με κλειδί k, το αφαιρεί από τον M και επιστρέφει την τιμή του, αλλιώς επιστρέφει null
- size(), isEmpty()

Παράδειγμα

Operation	*Output*	*Map*
isEmpty()	**true**	Ø
put(5,*A*)	**null**	(5,*A*)
put(7,*B*)	**null**	(5,*A*),(7,*B*)
put(2,*C*)	**null**	(5,*A*),(7,*B*),(2,*C*)
put(8,*D*)	**null**	(5,*A*),(7,*B*),(2,*C*),(8,*D*)
put(2,*E*)	*C*	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
get(7)	*B*	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
get(4)	**null**	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
get(2)	*E*	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
size()	4	(5,*A*),(7,*B*),(2,*E*),(8,*D*)
remove(5)	*A*	(7,*B*),(2,*E*),(8,*D*)
remove(2)	*E*	(7,*B*),(8,*D*)
get(2)	**null**	(7,*B*),(8,*D*)
isEmpty()	**false**	(7,*B*),(8,*D*)

Ένας απλός χάρτης βασισμένος σε λίστα

- Μπορούμε να υλοποιήσουμε έναν χάρτη με μία αταξινόμητη λίστα.
  - Αποθηκεύουμε τα στοιχεία του χάρτη σε μία λίστα S (βάσει μίας διπλά συνδεδεμένης λίστας), σε τυχαία σειρά



Απόδοση χάρτη που είναι βασισμένος σε λίστα

- Απόδοση:
  - Η put απαιτεί χρόνο $O(1)$ αφού μπορούμε να προσθέσουμε ένα νέο στοιχείο στην αρχή ή στο τέλος της ακολουθίας
  - Η get και η remove απαιτούν χρόνο $O(n)$ αφού στην χειρότερη περίπτωση (όπου το στοιχείο δεν υπάρχει) θα διασχίσουν ολόκληρη την ακολουθία ψάχνοντας ένα στοιχείο με το δεδομένο κλειδί
- Η υλοποίηση βάσει μη ταξινομημένης λίστας είναι κατάλληλη μόνο για χάρτες μικρού μεγέθους ή για χάρτες όπου η συνήθης λειτουργία είναι η put ενώ οι αναζητήσεις (get) και οι διαγραφές (remove) είναι σπάνιες (π.χ. ιστορικό καταγραφής συνδέσεων χρηστών σε έναν Η/Υ)